FWD - Bug #3325

Usage of ACCUMULATE and ACCUM with local buffer inside function makes translated code non-compilable

08/13/2017 09:29 AM - Jaroslaw Haziak

Status:	Test	**Start date:**	08/13/2017
Priority:	High	**Due date:**	
Assignee:	Eric Faulhaber	**% Done:**	90%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 08/13/2017 09:36 AM - Jaroslaw Haziak

The following code::

```
DEF INPUT PARAM inParam AS INT NO-UNDO.
DEF OUTPUT PARAM ouResult AS INT NO-UNDO.

FUNCTION test-func RETURNS INT (INPUT ipParam AS INT):

  DEF BUFFER bf_TestDoc FOR TestDoc.

  DEF VAR vCnt AS INT NO-UNDO.

  FOR EACH bf_TestDoc NO-LOCK
     WHERE bf_TestDoc.DocID >= inParam
     :
     ACCUMULATE bf_TestDoc.DocID (COUNT).
  END.

  RETURN ACCUM COUNT bf_TestDoc.DocID.

END.

ouResult = test-func(inParam).
```

is translated to the following non-compilable code:

```
public class TestProc3
{
   integer inParam;

   FieldReference fieldRef0 = new FieldReference(bfTestdoc, "docid");

   CountAccumulator accumCount0 = new CountAccumulator();

   /**
    * External procedure (converted to Java from the 4GL source code
    * in wd_test/test.proc3.p).
    */
   public void execute(final integer _inParam, final integer ouResult)
   {
      integer inParam = TypeFactory.initInput(_inParam);

      externalProcedure(TestProc3.this, new Block((Init) () ->
      {
         TestProc3.this.inParam = inParam;
```

```
         TypeFactory.initOutput(ouResult);
      },
      (Body) () ->
      {
         ouResult.assign(testFunc(inParam));
      }));
   }

   public integer testFunc(final integer _ipParam)
   {
      integer ipParam = TypeFactory.initInput(_ipParam);
      Testdoc.Buf bfTestdoc = RecordBuffer.define(Testdoc.Buf.class, "xdb", "bfTestdoc", "bf_TestDoc");

      return function("test-func", integer.class, new Block((Body) () ->
      {
         AdaptiveQuery query0 = new AdaptiveQuery();

         forEach("loopLabel0", new Block((Init) () ->
         {
            RecordBuffer.openScope(bfTestdoc);
            query0.initialize(bfTestdoc, "bfTestdoc.docid >= ?", null, "bfTestdoc.docid asc", new Object[]
            {
               inParam
            }, LockType.NONE);
            query0.addAccumulator(accumCount0);
            accumCount0.reset();
         },
         (Body) () ->
         {
            query0.next();
            accumCount0.accumulate();
         }));

         returnNormal(accumCount0.getResult());
      }));
   }
}
```

The reason is that new FieldReference(bfTestdoc, "docid") is put to the top-level scope where bfTestdoc is not accessible (because it is defined locally in testFunc method).

#2 - 08/24/2017 12:12 PM - Eric Faulhaber

- Status changed from New to WIP

- Assignee set to Eric Faulhaber

Created task branch 3325a for work on this issue.

#3 - 08/25/2017 12:36 AM - Eric Faulhaber

- File 3325_20170824a.patch added

- Status changed from WIP to Test

The attached patch fixes this testcase. Jaroslaw, please confirm that this works for your code.

A version of this fix was committed to task branch 3325a.

#4 - 08/25/2017 01:25 AM - Eric Faulhaber

- Status changed from Test to WIP

The fix has failed regression testing. I will keep investigating.

#5 - 09/21/2017 12:52 PM - Eric Faulhaber

Constantin, I've addressed the root cause of this issue by moving the scope of all Accumulator instances (and the fields and expressions they reference) to emit as data members of the Java block corresponding to the nearest enclosing, top-level 4GL block. This means they are declared/defined just inside the Java methods for the external procedure, internal procedure, function, etc. in which they are used.

In some cases, however, this has regressed the special IF statement and looping block processing you added (e.g., the calls to reset, postponeUnknown, etc.). These calls (and the try-finally they are in) have been dropped unintentionally with my changes. I think this is due to my removal of the promote-accum annotation in certain cases, but I'm tracking that down to be sure.

What is the significance of the promote-accum annotation w.r.t. this particular processing? For example, rules/convert/accumulate.rules checks for this in a while block with the following condition:

```
            pref.isRoot() == false and (targetref.isAncestorOf(0, pref) or
            targetref.id == pref.id or (getNoteBoolean("promote_accum") != null
            and getNoteBoolean("promote_accum")))
```

I'm trying to figure out if some other annotation can be used for this purpose (and where and under what conditions I would set it to achieve the desired effect), since promote-accum has the undesirable effect of making the Accumulator an instance variable of the top-level class, which may be out of the scope of a buffer it references (i.e., the root cause of this issue).

Also, can you please point me at the test case(s) you used originally to implement this feature set?

#6 - 09/21/2017 01:09 PM - Constantin Asofiei

See the testcases/uast/accum_tests_if*.p and other accum*.p tests.

For the other questions, I'll have to think a little more...

#7 - 09/21/2017 07:01 PM - Eric Faulhaber

Constantin Asofiei wrote:

> See the testcases/uast/accum_tests_if*.p and other accum*.p tests.

Thanks.

> For the other questions, I'll have to think a little more...

Don't worry about it, I think I have a fix for the regression. Testing it now.

#8 - 09/22/2017 03:29 AM - Eric Faulhaber

After another round of regression fixes, 3325a rev 11167 finally has passed conversion regression testing. I am running runtime regression testing now and I expect to run further runtime tests (ETF) tomorrow.

#9 - 09/22/2017 01:23 PM - Eric Faulhaber

- Status changed from WIP to Test

- % Done changed from 0 to 90

Task branch 3325a, rev 11167 passed regression testing (runtime and conversion) and was merged to trunk as rev 11165.

FWD 3.1, which we expect to release in the next few days, will include this fix. Although I have confirmed it fixes the test case provided by the original poster I am putting this issue in test status until he has a chance to obtain the code and confirm it fixes the underlying issue.

#10 - 10/03/2017 03:33 PM - Greg Shah

The fix for this task is included in FWD v3.1 which was recently released. Download it from
https://proj.goldencode.com/projects/p2j/wiki/FWD_v3_1_0

Please let us know if this issue can be closed.

Files